IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION

FOR

INTERPROCESS COMMUNICATION MECHANISM FOR HETEROGENEOUS COMPUTER PROCESSES

INVENTOR:

DAWSON F. DEAN

PREPARED BY:

JAMES D. IVEY
LAW OFFICES OF JAMES D. IVEY
5638 THORNHILL DRIVE
OAKLAND, CALIFORNIA 94611-2143
(510) 339-3944

FILE NUMBER: P-2057/723

Certificate of Mail by Express Mail under 37 CFR § 1.10

EXPRESS MAIL LABEL NO.: EI 261 597 477 US
Date of Deposit: July 14, 1997
I hereby certify that this paper or fee is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 CFR § 1.10 on the date indicated above and is addressed to Box: PATENT APPLICATION, Assistant Commissioner for Patents, Washington, D.C. 20231

James D. Ivey

SPECIFICATION

FIELD OF THE INVENTION

The present invention relates to computer software and, in particular, to a safe interprocess communication mechanism for communication with isolated computer processes.

BACKGROUND OF THE INVENTION

One of the latest, most exciting advances in popular computing is the advent and growth of the Internet and, in particular, the World Wide Web. The World Wide Web supports inter-related multimedia documents in which users can select, using conventional graphical user interfaces, which related document to display next. Such multimedia documents can include text, graphics, audio, and/or motion video. Within the sphere of Internet computing, one of the most exciting and promising developments is the development of applets written, for example, in the Java programming language developed by Sun Microsystems, Inc. of Mountain View, California. By supporting applets in the World Wide Web, active documents, i.e., documents which cause processing in a local computer when viewed, are added to the types of multimedia documents which can be accessed and viewed through the World Wide Web and other network protocols.

An applet is a collection of computer instructions which are designed to be transported through a computer network and executed within an applet viewing process executing within a local computer system. The computer system is local from the perspective of a user who requests retrieval and execution of the applet. Applets, by their very nature, raise security issues for such local computer systems. In general, computer programs can be configured, intentionally or inadvertently, to cause harm to the local computer system. Such harm can include, for example, erasing contents of persistent storage devices such as magnetic and optical disks, writing to sensitive areas of memory which are necessary for the proper functioning of the local computer system, and searching storage for sensitive information such as passwords and passing such

information to a recipient elsewhere on the computer network.

Applet viewing processes are sometimes referred to herein as applet viewers. Applet viewers, such as the Netscape Navigator World Wide Web viewer available from Netscape Corporation of Mountain View, California, generally prevent certain types of behavior of an applet to prevent harm from execution of the applet. For example, applets are prevented by the applet viewer from writing data to any persistent storage, thus protecting current contents of the persistent storage. In addition, some applet viewers perform array bounds checking and memory address checking to ensure that an applet does not read data from or write data to a portion of memory to which the applet should not have access. Similarly, while general purpose programming languages allow mathematic operations to be performed on memory address pointers for added flexibility, such is generally not permitted of applets since such makes checking memory access for unauthorized addresses particularly difficult.

Some applet viewers provide a virtual machine in which the applet executes. The virtual machine includes a virtual processor which executes computer instructions from an instruction set, and applets are constructed of computer instructions of that instruction set. In executing the applet, the computer instructions of the applet are translated into a native instruction set capable of execution by the physical processor of the local computer system and then executed by the physical processor. In addition, the virtual machine has a virtual memory space which is used by the applet as if the virtual memory space was a physical memory space accessed by a conventional computer process. The applet viewer translates addresses of the virtual memory space into addresses in a physical memory space of the local computer system and effects memory access by the applet in the physical memory space. The applet viewer can therefore prevent the applet from gaining access to sensitive information stored in the physical memory space of the local computer system. The result of the virtual machine is that the applet perceives itself to be the sole computer process within a computer, namely, the virtual machine. The applet is therefore intentionally isolated by the applet viewer such that execution of the applet cannot interfere with the proper functioning of the local computer system and other computer processes executing therein.

One disadvantage of such isolation of applets is that other computer processes executing

concurrently with and independently of the applet viewer cannot communicate with applets executing in the virtual machine. A computer process is a collection of computer instructions which define a task to be performed by a computer and an execution state which includes, for example, an address space for storing data which can be manipulated by execution of one or more of the computer instructions. Many computer programs in use today are not implemented as a single computer process but as multiple computer processes which communicate with one another through conventional interprocess communication mechanisms. In addition, many computer processes serve as server computer processes to receive processing requests from client computer processes and to process such requests. Thus, one attribute of a computer process which is currently highly desirable in certain computer processing environments is the attribute of responsiveness to processing requests of another computer process.

Applets are generally prevented from using any conventional means for receiving and responding to computer processing requests from other computer processes. Many conventional interprocess communication protocols operate at a low level, i.e., by direct manipulation of computer system resources. As described above, such direct manipulation of resources of the client computer system is typically disallowed by the applet viewer. Higher level protocols for interprocess communication would require modification of the applet viewer to implement such an interface with the applet. However, the developer of the applet and the developer of the applet viewer are most frequently separate entities such that modification of the applet viewer by the developer of the applet itself is impractical.

What is needed is an interprocess communication mechanism in which applets can receive and respond to processing requests of other computer processes and can send processing requests to such other computer processes without requiring modification of applet viewers. In addition, the interprocess communication mechanism should not submit the local computer system to undue risks of harm from the applets whether the applets are malicious or negligent.

SUMMARY OF THE INVENTION

In accordance with the present invention, interprocess communication between a computer process and an applet executing within an applet viewer is achieved by encoding remote procedure calling (RPC) requests as requests for documents in a known, standard document request format, such as a hypertext transfer protocol (HTTP) universal resource locator (URL). As used herein, a document is a data file which can store a collection of data such as data representing text, one or more graphical images, audio sounds, and/or motion video. A portion of the name space for documents which can be retrieved in HTTP is reserved for RPC requests. An applet encodes an RPC request as a request to receive a document in the portion of the name space reserved for RPC requests and sends the URL to an RPC process. The RPC process includes an HTTP server and (i) receives the URL; (ii) determines that the URL specifies a document in the name space portion reserved for RPC requests; (iii) parses the RPC request from the URL; and (iv) services the RPC request. In addition, the RPC process places any results produced by servicing the RPC request into a document which is then sent to the applet.

By using a known, standard document retrieval mechanism such as HTTP URLs, RPC interprocess communication can be implemented in isolated computer processes such as applets executing in an applet viewer. Specifically, one of the few things an applet viewer permits an applet to do is request a document according to a known, trusted document retrieval protocol such as HTTP. From the perspective of the applet viewer, the RPC request is no more than an HTTP URL which requests retrieval of a document according to the hypertext transfer protocol and is therefore permitted. In addition, since the results of the execution of the requested RPC function are sent to the applet as a document in accordance with HTTP, the applet viewer perceives the receipt of the results as no more than the receipt of the requested document according to HTTP. In addition, since the applet is still denied direct access to computer system resources, computer system security is preserved.

Further in accordance with the present invention, an applet can make itself available to serve RPC requests from an independently executing computer process. Specifically, the applet encodes as an HTTP URL data indicating that the applet is available to process RPC requests. As

described above, the applet viewer permits the applet to send HTTP URL requests to computer processes which implement an HTTP server. In response to the URL, the computer process begins sending a virtual document to the applet. Sending of the virtual document is suspended until the computer process has a processing request for the applet. The computer process builds an RPC request representing the processing request and sends the RPC request to the applet as a portion of the virtual document sent in response to the URL received from the applet. Upon receipt of the portion of the virtual document, the applet parses the RPC request and services the RPC request. Accordingly, the applet can serve RPC requests of independent computer processes in a manner permitted by an applet viewer without compromising computer system security.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a computer system which includes an applet, an applet viewer, and an RPC computer process in accordance with the present invention.

Figure 2 is a block diagram showing the applet and RPC computer process of Figure 1 in greater detail.

Figure 3 is a logic flow diagram of the processing of the initiation of an RPC request by the applet of Figure 2.

Figure 4 is a logic flow diagram of the processing of the RPC request by the RPC computer process of Figure 2.

Figure 5 is a logic flow diagram of the parsing of the RPC request by the RPC computer process of Figure 2.

Figure 6 is a logic flow diagram of the servicing of the RPC request by the RPC computer process of Figure 2.

Figure 7 is a logic flow diagram of the manner in which the applet of Figure 2 makes itself available to process RPC requests from the RPC computer process of Figure 2.

DETAILED DESCRIPTION

In accordance with the present invention, remote procedure calling (RPC) requests are encoded as requests for documents in a known, standard document request format, such as an hypertext transfer protocol (HTTP) universal resource locator (URL). A portion of the name space for documents which can be retrieved in HTTP is reserved for RPC requests. An applet 200 (Figure 1) encodes an RPC request as a request to receive a document in the portion of the name space reserved for RPC requests and sends the URL to an RPC process 210. RPC process 210 receives the URL and determines that the URL specifies a document in the name space portion reserved for RPC requests and parses the RPC request from the URL and services the RPC request. In addition, RPC process 210 places any results produced by servicing the RPC request into a document which is then sent to applet 200. In this way, applet 200 communicates RPC requests to RPC process 210 in a manner generally permitted by applet viewer 150 within which applet 200 executes. In addition, applet 200 can make itself available to receive RPC requests from RPC process 210 in a manner generally permitted by applet viewer 150 as described more completely below.

In the illustrative embodiment described herein, applet 200 is an applet which executes within applet viewer 150 which is part or all of a computer process executing within a computer system 100, and RPC process 210 is part or all of a computer process executing within a computer system 100.

Computer system 100 includes a processor 102 and memory 104 which is coupled to processor 102 through an interconnect 106. Interconnect 106 can be generally any interconnect mechanism for computer system components and can be, for example, a bus, a crossbar, a mesh, a torus, or a hypercube. Processor 102 fetches from memory 104 computer instructions of a computer process such as applet viewer 150 and executes the fetched computer instructions. In addition, processor 102 can fetch computer instructions through computer network 140 through network access circuitry 160 such as a modem or ethernet network access circuitry. Processor 102 also reads data from and writes data to memory 104 and sends data and control signals

through interconnect 106 to one or more computer display devices 120 and receives data and control signals through interconnect 106 from one or more computer user input devices 130 in accordance with fetched and executed computer instructions.

Memory 104 can include any type of computer memory and can include, without limitation, randomly accessible memory (RAM), read-only memory (ROM), and storage devices which include storage media such as magnetic and/or optical disks. Memory 104 stores applet viewer 150 and RPC process 210 which are computer processes which execute concurrently and independently within processor 102 from memory 104.

Each of computer display devices 120 can be any type of computer display device including without limitation a printer, a cathode ray tube (CRT), a light-emitting diode (LED) display, or a liquid crystal display (LCD). Each of computer display devices 120 receives from processor 102 control signals and data and, in response to such control signals, displays the received data. Computer display devices 120, and the control thereof by processor 102, are conventional.

Each of user input devices 130 can be any type of user input device including, without limitation, a keyboard, a numeric keypad, or a pointing device such as an electronic mouse, trackball, light-pen, touch-sensitive pad, digitizing tablet, thumb wheels, or joystick. Each of user input devices 130 generates signals in response to physical manipulation by a user and transmits those signals through interconnect 106 to processor 102.

RPC process 210 and applet 200 are described in greater detail in the context of Figure 2. Applet 200 is configured to invoke RPC functions, e.g., either of RPC functions 206A-B of RPC process 210, to thereby incorporate the tasks performed by such RPC functions into a larger task performed by applet 200. RPC process 210 includes an HTTP server 204 which serves HTTP requests in a conventional manner, i.e., receives a URL which specifies a requested document and produces the requested document in response to the received URL. RPC process 210 also includes a URL filter 202 and RPC functions 206A-B. URL filter 202 reserves a portion of the name space of documents which can be requested using a URL for RPC requests. As described more completely below, URL filter 202 determines whether a particular URL specifies a

document in the reserved name space portion and processes the URL according. In accordance with HTTP, applet 200 sends a URL specifying a document to RPC process 210 and receives the specified document from RPC process 210. To invoke either of RPC functions 206A-B, applet 200 forms a URL according to the steps of logic flow diagram 300 (Figure 3) and sends the URL to RPC process 210.

In step 302, applet 200 (Figure 2) selects a function identifier, i.e., data which uniquely identifies either RPC function 206A or RPC function 206B. In one embodiment, the function identifier is alphanumeric. In step 304 (Figure 3), applet 200 (Figure 2) selects zero or more arguments to supply to the identified RPC function as input data. The particular RPC function identified and the particular arguments to be provided to the identified RPC function are determined according to the particular task to be performed by applet 200 in accordance with the design and implementation of applet 200. In step 306 (Figure 3), applet 200 (Figure 2) encodes the function identifier and arguments as a URL request in the reserved name space portion to thus encode the function identifier and arguments as an RPC request.

RPC process 210 receives the encoded URL in step 402 (Figure 4) of logic flow diagram 400 and processes the URL, and all other URLs, according to logic flow diagram 400. Processing transfers to test step 404 in which URL filter 202 (Figure 2) of RPC process 210 determines whether the URL is an encoded RPC request, i.e., whether the received URL specifies a document in the reserved name space portion. The entire name space for all URLs includes (i) an identifier for a particular host, which can specify server computer system 102 (Figure 1), for example, (ii) a port through which the host is accessed, (iii) one or more embedded directories, and (iv) a file name. The portion of the name space which is reserved for RPC requests can be any of the following or any combination of the following: (i) a specific host identifier, (ii) one or more specific directories, or (iii) a specific pattern in the file name. In one embodiment, the reserved name space portion is a particular directory such that URLs specifying the particular directory, or any subdirectory thereof, are recognized by URL filter 202 as RPC requests. In test step 402 (Figure 4), URL filter 202 (Figure 2) determines whether the received URL is an RPC request by comparing one or more components of the URL to specific component values reserved

from the URL name space for RPC requests.

If URL filter 202 (Figure 2) determines that the received URL is an RPC request, processing transfers to step 410 (Figure 4) which is described more completely below. Conversely, if URL filter 202 (Figure 2) determines that the received URL is not an RPC request, processing transfers to step step 406 (Figure 4). In step 406, URL filter 202 (Figure 2) forwards the received URL to HTTP server 204 which processes the URL in a conventional manner, i.e., retrieves the document specified by the URL and provides the specified document to URL filter 202.

In step 410 (Figure 4), to which processing transfers from test step 404 if URL filter 202 (Figure 2) determines that the received URL is an RPC request, URL filter 202 (Figure 2) parses the received URL into an RPC request. Step 410 (Figure 4) is shown in greater detail as logic flow diagram 410 (Figure 5). In step 502, URL filter 202 (Figure 2) parses the RPC function identifier from the received URL. The following is an illustrative example of a URL representing an RPC request.

http://serverhost:7123/function=function.name&arg1=arg1.data&arg2=arg2.data&arg3=arg3.data (1)

In URL (1), "http" indicates that URL specifies a document to be retrieved according to HTTP. While the hypertext transfer protocol (HTTP) is described in the illustrative example described herein, it is appreciated that other protocols which allow for data transfer can be used. Protocols which are widely implemented and supported, secure, and largely trusted are preferred. For example, the known file transfer protocol (FTP), Gopher, simple mail transfer protocol (SMTP), and Internet relay chat (IRC) are protocols in which RPC requests can be encoded as document requests. "Serverhost" identifies computer system 100 (Figure 1) as the host to which the URL is directed through network 140. "7123" identifies a port of computer system 100.

The remainder of URL (1) syntactically specifies a file within computer system 100 that is to be retrieved through a logical port whose identifier is 7123. However, the remainder of URL

(1) in effect specifies an RPC function and provides three arguments to be used as input data by the specified RPC function. Specifically, "function=function.name" identifies one of RPC functions 206A-B (Figure 2) as the RPC function to be executed on behalf of applet 200. In addition, "arg1=arg1.data&arg2=arg2.data&arg3=arg3.data" specifies that the alphanumeric data strings "arg1.data," "arg2.data," and "arg3.data" are to be supplied to the identified RPC function as input data. It is appreciated that "function=function.name&arg1=arg1.data&arg2=arg2.data&arg3=arg3.data" can be an invalid file specification within computer system 100 (Figure 1) and can therefore identify no valid document within computer system 100. However, URL filter 202 (Figure 2) can be configured to accept as RPC requests URLs which cannot identify an actual document stored within computer system 100 (Figure 1). All that is important is that the specification of the RPC function and arguments comports with the syntax of a HTTP URL such that applet viewer 150 permits the RPC request encoded as a URL to be sent to RPC process 210.

In step 502 (Figure 5), URL filter 202 (Figure 2) parses "function.name" from URL (1) to determine which of RPC functions 206A-B to invoke. In this illustrative example, "function.name" identifies RPC function 206A. In step 504 (Figure 5), URL filter 202 (Figure 2) parses the alphanumeric data strings "arg1.data," "arg2.data," and "arg3.data" from URL (1). After step 504 (Figure 5), processing according to logic flow diagram 410, and therefore step 410 (Figure 4), completes.

In step 412, URL filter 202 (Figure 2) of applet 200 services the RPC request parsed from the URL received in step 402 (Figure 4). Step 412 is shown in greater detail as logic flow diagram 412 (Figure 6) in which processing begins in step 602. In step 602, URL filter 202 (Figure 2) invokes execution of the RPC function identified by the URL received in step 402 (Figure 4), e.g., RPC function 206A (Figure 2) identified by URL (1) above. In step 604 (Figure 6), URL filter 202 (Figure 2) supplies the arguments parsed in step 504 (Figure 5) to the identified RPC function as input data. As a result, the identified RPC function, e.g., RPC function 206A (Figure 2), performs the task requested by applet 200. After step 604, processing according to logic flow diagram 412, and therefore step 412 (Figure 4), completes.

Processing transfers to step 414 in which URL filter 202 (Figure 2) receives from RPC function 206A any results produced by execution of RPC function 206A and packages those results into a document. Processing transfers from either step 414 (Figure 4) or step 406 to step 408. In step 408, URL filter 202 (Figure 2) sends a document to applet 200. The document includes the results as packaged in step 414 (Figure 4) if the received URL is an RPC request or is the document requested by the received URL otherwise.

Thus, according to the present invention, an RPC request is encoded as a URL which appears to request a document according to the hypertext transfer protocol. Many applet viewers, such as applet viewer 150, execute applets such as applet 200 and allow such applets to send requests for documents according to standardized, trusted protocols and allow such applets to receive documents in response to such requests. As a result, encoding RPC requests as URLs which appear to request a document according to HTTP allows an applet executing within an applet viewer to send RPC requests to processes executing independently of the applet viewer notwithstanding isolation imposed upon such applets by applet viewers.

Receipt and Processing of RPC Requests by Applet 200

Applet 200 (Figure 2) can make itself available to receive RPC requests from RPC process 210 in a manner which is generally permitted by applet viewer 150 (Figure 1) and which is illustrated in logic flow diagram 700 ((7). Processing according to logic flow diagram 700 begins in step 702.

In step 702, applet 200 ((2) builds an RPC request for execution of an "RPC ready" RPC function by RPC process 210 and encodes the RPC request as a URL in the manner described more completely above. In step 704 ((7), applet 200 ((2) sends the URL encoded in step 702 ((7) to RPC process 210 to thereby request execution of the "RPC ready" RPC function, which can be RPC function 206B, for example.

The design and implementation of RPC function 206B is such that execution thereof indicates to RPC process 210 that applet 200 is ready to receive RPC requests from RPC process 210 and establishes a communications channel through which RPC process 210 can send RPC

requests to applet 200. Specifically, HTTP, as implemented by both RPC process 210 and applet viewer 150 ((1) within which applet 200 executes, expects a document to be retrieved in response to the URL sent in step 704 ((4). In addition, HTTP as implemented permits transfer of the requested document to be delayed and intermittent. However, to RPC process 210 requests a virtual document, i.e., a document which does not exist within memory 104 ((1) of computer system 100 but which is instead created in response to the URL. Execution of RPC function 206B ((2) of RPC process 210 changes the state of RPC process 210 to indicate that applet 200 is ready to receive RPC requests and to store data identifying the communications channel through which applet 200 is waiting to receive a document in response to the URL sent in step 704 ((7).

RPC process 210 includes a core function 208 which defines and implements a central task for which RPC process 210 is designed. Execution of core function 208 can include sub-tasks which are implemented by one or more of RPC functions 212 of applet 200. Accordingly, to cause performance of such sub-tasks, core function 208 of RPC process 210 builds RPC requests which request execution of a selected one of RPC functions 212 and includes zero or more parameters to be used by the selected RPC function as input data. To send such an RPC request to applet 200, RPC process 210 sends the RPC request to applet 200 as a portion of the virtual document requested by the URL sent by applet 200 in step 704 ((7). By sending the RPC request as only a portion of the requested virtual document, RPC process 210 ((2) indicates to applet 200 that other RPC requests can be subsequently sent to applet 200 through the same communication channel. Since the RPC request is sent to applet 200 as part of a document, the contents of which are not constrained by any particular protocol such as HTTP, the RPC request can be in any convenient form and can be in a form which is entirely inappropriate for a HTTP URL. To terminate the communication channel, and therefore terminate the ability of applet 200 to receive RPC requests from RPC process 210, RPC process 210 sends data indicating that the entirety of the virtual document requested by applet 200 has been sent to applet 200.

Processing by applet 200 transfers from step 704 ((7) to loop step 706 in which steps 708-712 are performed repeatedly until applet 200 ((2) receives data indicating that the entirety of the requested virtual document has been received. In step 708 ((7), applet 200 ((2) receives a portion

of the virtual document from RPC process 210. In step 710 ((7), applet 200 ((2) parses an RPC request from the received portion. As described above, the format of the RPC request can be entirely independent of the format of HTTP URLs. In step 712 ((7), applet 200 ((2) services the parsed RPC request by executing the one of RPC functions 212 specified by the parsed RPC request and supplying any arguments parsed from the received portion as input data. Any results produced by servicing the parsed RPC request can be communicated to RPC process 210 in the form of an HTTP URL built and sent to RPC process 210 in the manner described above. The results URL identifies the one of RPC functions 212 invoked by the parse RPC request to specify to RPC process 210 to which RPC request the resulting data pertains.

Steps 708-712 ((7) are repeated until applet 200 ((2) receives data from RPC process 210 indicating that the entirety of the requested virtual document has been sent to applet 200. Thereafter, processing according to logic flow diagram 700 completes.

In this way, applet 200 accepts RPC requests from RPC process 210 in a manner which is permitted by applet viewer 150 ((1) without requiring modification of applet viewer 150. Accordingly, many of the advantages of interprocess communication are achieved in the secure context of an applet viewer.

The above description is illustrative only and is not limiting. The present invention is limited only by the claims which follow.